Max/MSP: guía de programación para artistas

Max 6: Interfaz

Francisco Colasanto

CMMAS.org
Centro Mexicano para la Música y las Artes Sonoras

Primera edición, 2012

Casa de la Cultura (planta alta)
Av. Morelos Norte No. 485.
Centro. Morelia, Michoacán, México. C.P. 58000
www.cmmas.org
info@cmmas.org

ISBN: en trámite

Hecho en México

Traducción de la versión en inglés: Roberto Becerra
Revisión de traducción al inglés: Tim Poulin
Diseño de tapa: Francisco Colasanto

Índice

Presentación

Esta propuesta novedosa de Francisco Colasanto sobre Max/MSP significa un paso más en los proyectos de creación de materiales de formación del Centro Mexicano para la Música y las Artes Sonoras (CMMAS) y de consolidación de los proyectos de investigación de Francisco. Hay varias cosas que vale la pena mencionar. En primera instancia podría identificarse como el resultado de la nueva etapa de trabajo del CMMAS, en donde intentamos actualizarnos y utilizar las tecnologías más novedosas para la creación, difusión y distribución de contenidos. De algún modo, en esta nueva publicación, todos en el CMMAS, hemos tenido una labor que describiría como "satelital", porque giramos permanentemente en torno al trabajo de Francisco, quien se ha convertido en eje central de la existencia y capacidad técnica del Centro. Los que estamos a su alrededor hemos visto crecer el proyecto que se había gestado incluso antes del nacimiento del CMMAS y hemos visto la tenacidad y convicción con la cual ha trabajado en este nuevo paso para extender el alcance de tan significativo proyecto. Este texto es entonces el resultado de un proceso tanto personal como profesional de mucho tiempo y tremendo empeño, que pasó de ser individual, a ser un proyecto de equipo. Así mismo, demuestra un compromiso pedagógico y de investigación serio y me atrevo a asegurar que es único en su tipo en América Latina y probablemente a nivel internacional. Muestra una capacidad de entender el trabajo del creador sonoro interesado en Max/MSP, diverso y enfocado en objetivos definidos que pretenden ofrecer herramientas útiles y directas para el artista.

Max/MSP: guía de programación para artistas, en su nueva etapa representa una muestra tangible de la preocupación de Colasanto por compartir su conocimiento sobre la herramienta más trascendente de los últimos años, con los compositores y con los artistas sonoros interesados en tecnología. Este texto complementa los esfuerzos editoriales del CMMAS de manera singular ya que

se convierte en el proyecto más ambicioso de formación e investigación. Con ello, el Centro pretende convertirse en una alternativa capaz de fomentar el uso inteligente y profesional de las herramientas más avanzadas. Las nuevas tecnologías ofrecen la capacidad adicional para interactuar y entender los pormenores de diferentes temas indispensables sobre Max/MSP y con ellos recorrer el aprendizaje de manera personalizada, detallada y tecnológicamente atractiva.

Este libro electrónico es, entonces, un proyecto de investigación técnica y pedagógica que ha logrado integrar un sistema de transmisión de conocimiento complejo de manera eficiente y bien sistematizado con la ventaja de que es posible comprender el texto a diferentes niveles de profundidad. Se puede utilizar como una fuente de referencia sobre el lenguaje de programación Max, el cual se ha convertido en indispensable para todos los artistas interesados en el control del audio, dispositivos MIDI y tecnologías de vanguardia actuales; pero al mismo tiempo estoy seguro que será un recurso invaluable para todos aquellos que estamos interesados en aprender paso a paso un lenguaje de programación que se ha instalado como central en la actualidad y experimentarlo de manera claramente vinculada con el proceso creativo. Con este texto Francisco ha logrado que todos podamos entender Max desde una perspectiva que nos sea útil como artistas. Con procedimientos, ejemplos y niveles de profundidad y complejidad que controla el lector y que permite entonces, repasar siempre que necesitemos repensar. El autor nos invita a compartir aproximaciones para resolver problemas sobre cómo trabajar el sonido, pero con una perspectiva de clara experiencia pedagógica y técnica que le da a este texto la autoridad que estoy seguro será demostrada con el tiempo.

Hoy en día, en un mundo de tecnología que se hace obsoleta al momento de nacer, nos preguntamos cual es el significado de un texto sobre una herramienta específica. Creo que esto ha sido claramente identificado por su autor desde el inicio, y el resultado es un compendio de estrategias y ejemplos actualizados sobre estructuras de pensamiento implementadas de forma que los cambios y mejoras en los equipos y versiones del programa no afecten de manera importante.

Digo esto porque la idea modular y de objetos que ha consolidado a Max/MSP como revolucionario, ha sido identificada como un método pedagógico eficiente con el cual el libro ofrece de manera simultánea la explicación y experimentación de los objetos y elementos específicos de Max, pero también una explicación general del por qué y cómo de cada procedimiento, objeto e idea. Es así como se respalda una forma de aprendizaje que trasciende versiones y modelos específicos. Colasanto busca enseñar Max/MSP pero además pretende enseñar a aprender Max/MSP.

Para el CMMAS es un paso central en la consolidación de las aspiraciones por convertirnos en un espacio de reflexión sobre lo que la tecnología y su implementación significan para el trabajo creativo. Es sin duda alguna, un escalón definitivo en nuestro intento por mostrar que en México puede haber un espacio de alcance internacional que puede impulsar la creación, la formación y la producción, al mismo tiempo que se mantiene como un Centro de carácter público. El CMMAS es entonces un espacio de vinculación y de transferencia de conocimiento a través de procesos creativos que sólo se pueden compartir a través de la experiencia y de la reflexión consciente sobre el significado e impacto de las nuevas tecnologías. Este proyecto es pieza fundamental para alcanzar este objetivo y estoy seguro que será parte importante de un cambio en la perspectiva educativa para los artistas hispanohablantes a los cuales el texto proveerá de la información necesaria para incorporarse, de manera preparada y crítica, al medio artístico tecnológico internacional.

No me queda más que agradecer como director del CMMAS, a las instituciones que han apoyado este proyecto, al personal del CMMAS por crear un espacio único de trabajo y compromiso con el crecimiento humano de todos los que aquí laboramos y de la comunidad de Michoacán a la que pertenecemos. Agradezco a Francisco por compartir con nosotros, de manera desinteresada y creativa, su conocimiento y experiencia que hoy quedan disponibles a través de los nuevos dispositivos móviles, en este trabajo que ha demostrado que atiende las necesidades de muchos artistas desde su primera edición impresa,

en 2010. Los invito a experimentar Max/MSP de manera diferente, amigable, clara y sobre todo útil. Tienen en sus manos la mejor herramienta sobre Max/MSP para apoyar en el uso de la mejor herramienta que existe: la creatividad.

Dr. Rodrigo Sigal
Director, CMMAS

Notas acerca de esta publicación

Esta publicación es una continuación y una extensión de "MaxMSP: guía de programación para artistas Vol I", por lo tanto, si bien está esta dirigida a todos aquellos artistas que deseen trabajar con Max/MSP, no es de carácter introductorio como el mencionado libro (con excepción de esta primera entrega gratuita).

Tiene como objetivo desarrollar diferentes áreas relacionadas con el procesamiento de señales digitales (DSP) y la creación sonora.

Su publicación se realizará en forma de módulos los cuales se ocuparán de un tema específico. El lector puede escoger adquirir todos los módulos o sólo aquellos que sean de su interés.

Max es un software desarrollado por Cycling74 (www.cycling74.com) . Desde su web es posible bajar el software que, una vez instalado, ofrece una versión demo de 30 días. Todos los ejemplos gráficos que aquí y en los sucesivos módulos se encuentran (fotografías, audio, video) pertenecen a ejemplos realizados en Max/MSP. Estos ejemplos podrán descargarse desde www.maxmsptutorial.com

Agradecimientos

Deseo agradecer muy especialmente a todos los que hicieron posible esta publicación: el Centro Mexicano para la Música y las Artes Sonoras (CMMAS), a su director el Dr. Rodrigo Sigal Sefchovich y a su personal; a Roberto Becerra por la traducción al inglés y a Tim Poulin por la revisión de dicha traducción.

También quisiera agradecer a todos aquellos que me apoyaron y aportaron sus ideas y asistencia.

Interfaz: Patcher

Al abrir Max 6 por primera vez, aparece una ventana llamada *Getting Started with Max6* (Figura 1) desde la cual podremos acceder a diferentes áreas que nos permitirán introducirnos a este software, es decir, que desde aquí podremos acceder a los foros de Max 6, documentación, videos, etc. También aquí podemos escoger empezar a trabajar en un nuevo patcher a través del primer recuadro (*Get Started*).

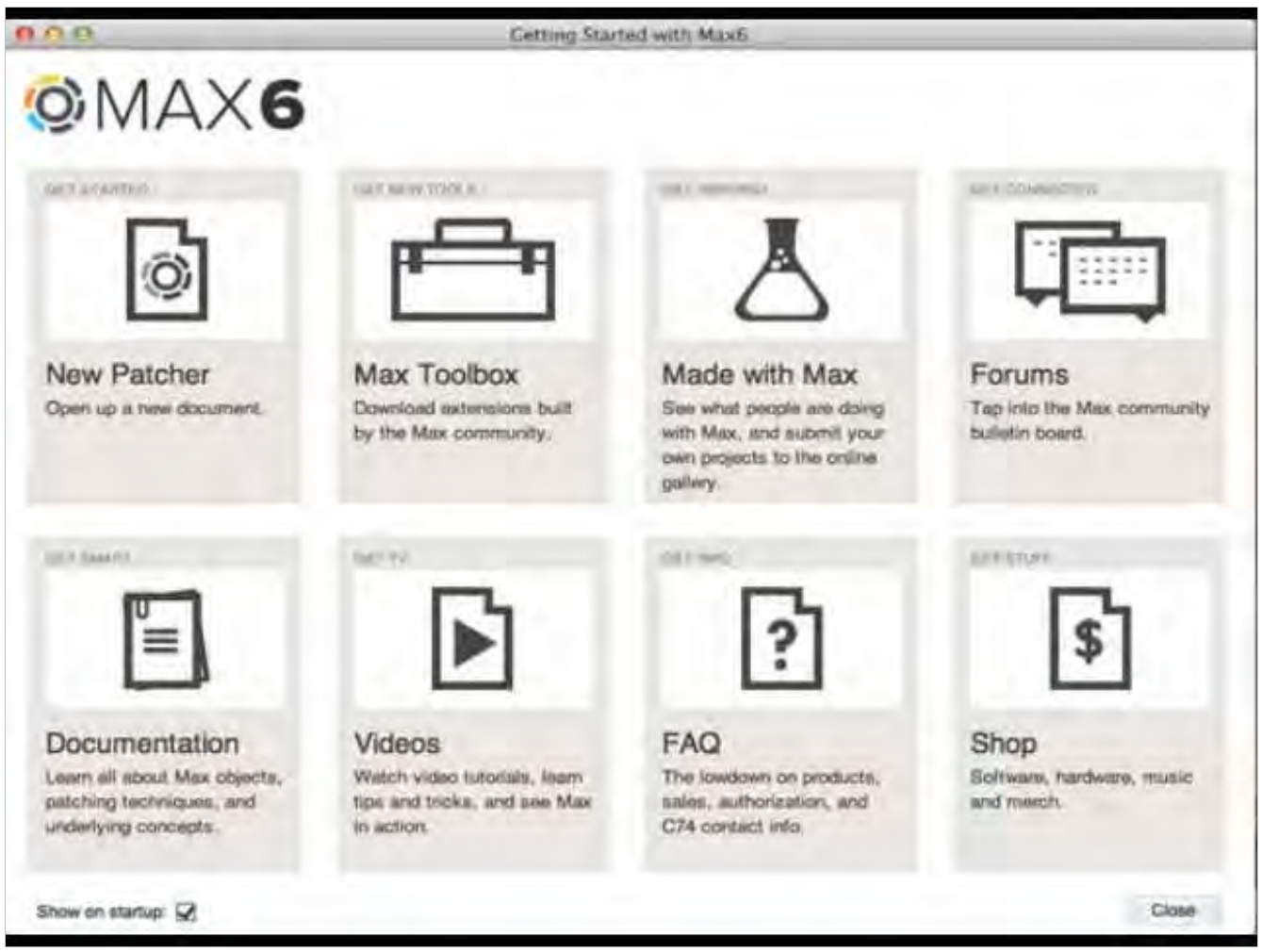


Figura 1

También podemos crear una nueva programación o patcher, desde *File -> New Patcher* (o a través del shortcut + N*). Cuando hacemos esto nos encontraremos con una ventana en blanco (ventana de patcher) como la que observamos en la figura 2 y en la cual podremos comenzar a colocar e interconectar una serie de objetos. A esta interconexión la llamamos patch.

* Este shortcut pertenece al sistema operativo de Macintosh. En Windows, por lo general, la tecla corresponde a Ctrl.

Figura 2

Si hacemos doble clic en cualquier parte del patcher nos aparecerá un menú de herramientas gráficas. Las herramientas gráficas o UI (User Interface) se encuentran agrupadas en diferentes categorías: Audio, Básicas (Basic), Botones (Buttons), etc. Para colocar una de estas UI en nuestro patcher, debemos hacer doble clic sobre la que escojamos o arrastrarla al patch.

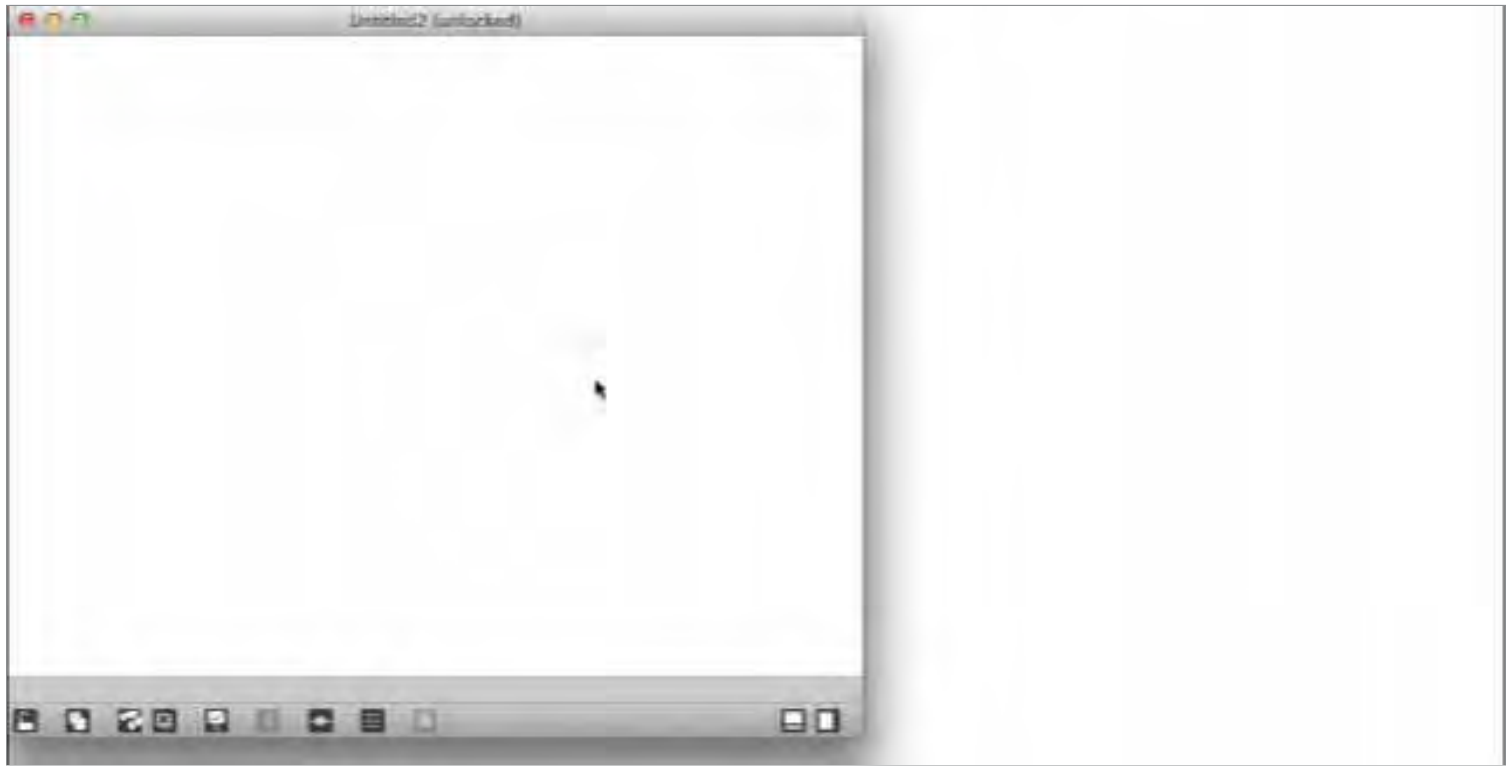


Figura 3

Otra forma de escoger una de estas herramientas es abriendo la barra que aparece en el costado derecho cuando hacemos clic sobre el botón que esta en la esquina derecha inferior de la ventana. Vemos esto en la figura 3.

Modo de edición / ejecución / presentación

Cuando queremos colocar nuevos objetos, moverlos dentro del patcher, o conectar un objeto con otro, trabajamos en el *modo de edición* o desbloqueado al cuál se accede desde el menú *View -> Edit* o + e en Macintosh y ctrl. + e en Windows. En cambio cuando queremos que esos objetos respondan al clic del mouse o interactuar con las herramientas gráficas, trabajamos en el *modo de ejecución* o bloqueado (otra vez *View -> Edit*, + e). Por ejemplo, si estamos en el *modo de edición*, será posible hacer doble clic sobre el patcher para que nos aparezca la ventana de herramientas y así poder tomar la interfaz de usuario llamada *slider*. Este objeto tiene la capacidad de funcionar como un fader virtual cuando hacemos clic sobre él y, manteniendo el botón izquierdo del mouse presionado, lo deslizamos horizontalmente (si esa es la orientación que le damos); sin embargo, si queremos hacer que ese *slider* funcione como tal, debemos cambiar del *modo de edición* al *modo de ejecución*, de lo contrario, al mover el mouse sólo moveremos de lugar el objeto. Lo mismo sucede con todas las herramientas gráficas que posee Max 6.

Para asegurarnos de estar trabajando en el modo deseado, podemos observar la leyenda que se encuentra junto al nombre del patcher en la parte superior de la ventana. Si dice "unlocked" significa que estamos en el *modo de edición*, de lo contrario nos encontramos en el *modo de ejecución*. Lo vemos en la figura 4.

Otro modo en el que podemos trabajar es el *modo de presentación*, que aparece con la versión 5 de Max. Permite cambiar la ubicación y el tamaño de los objetos independientemente de cómo se encuentren dispuestos en el *modo de ejecución*. Por ejemplo, en la figura 5, vemos un patch que se encuentra en el *modo*

de ejecución y pasa al *modo presentación*.

Figura 4

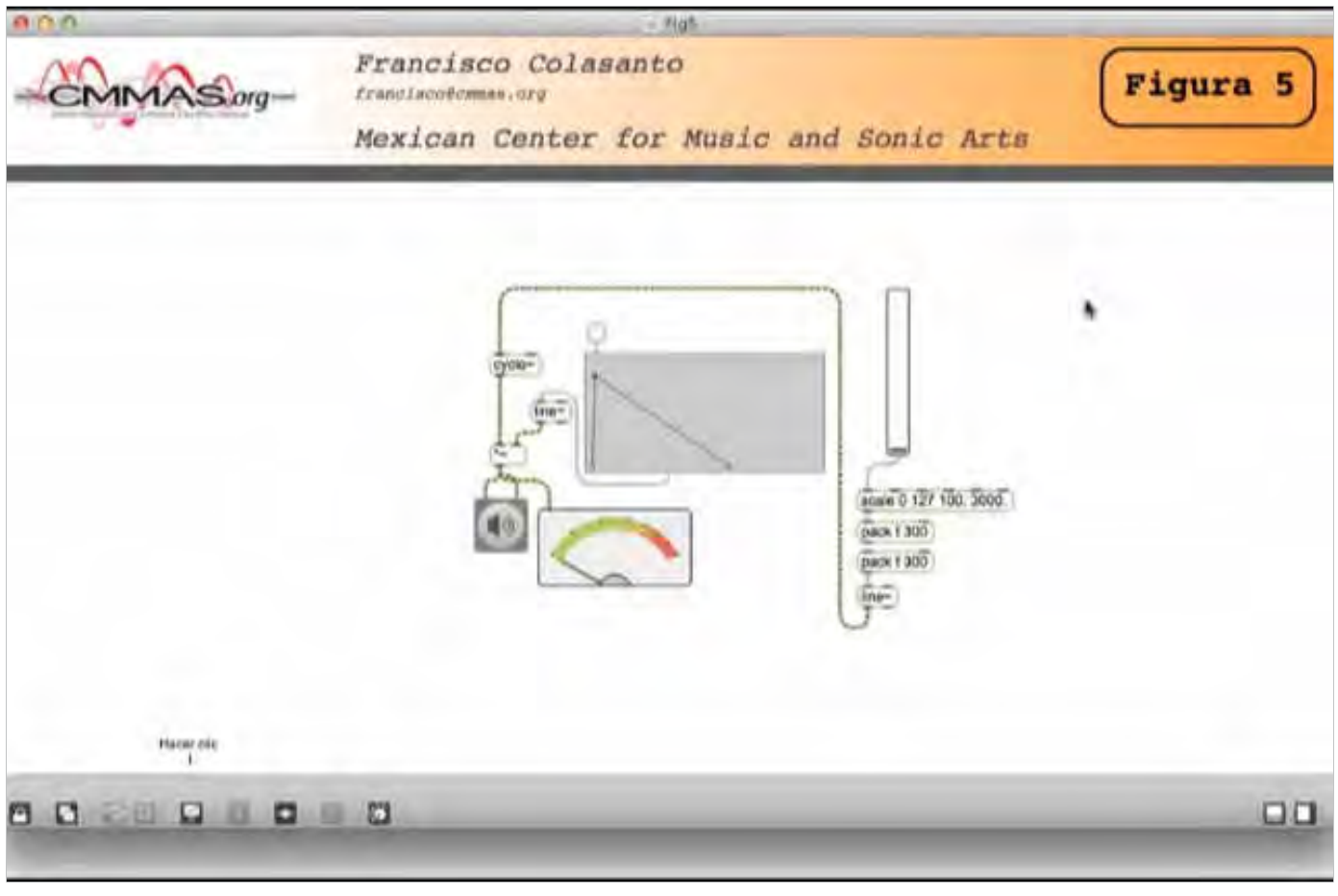
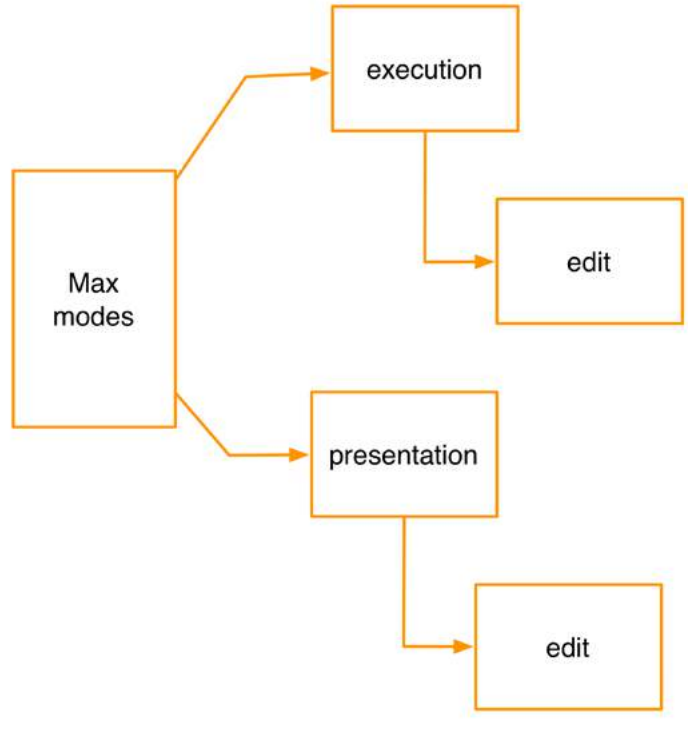
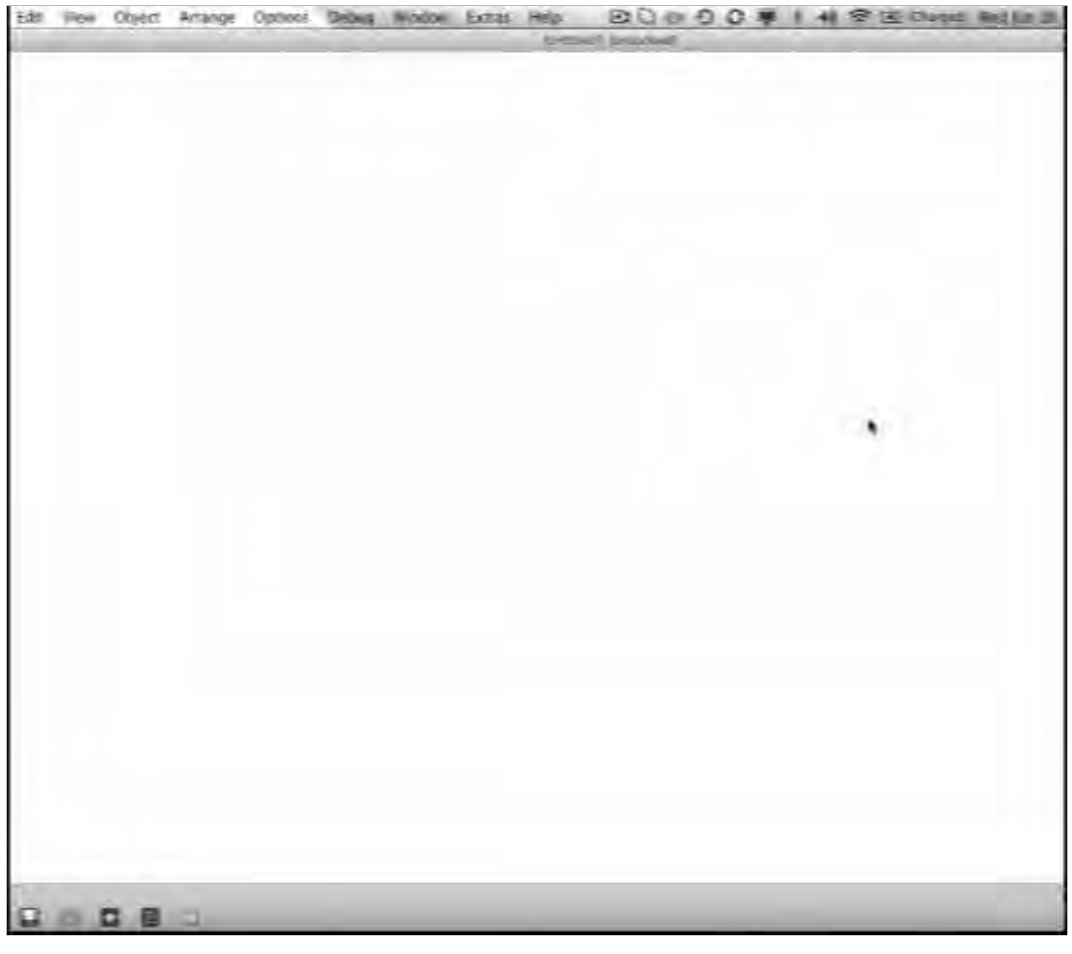


Figura 5

Tanto el *modo de ejecución* como el *modo de presentación* tienen un *modo de edición* propio. Para realizar el ejemplo de las figura 5, seleccionamos en el *modo de edición* del *modo de ejecución* los objetos que queremos que aparezcan en el *modo de presentación*. Veamos el siguiente cuadro que nos ayudará a aclarar lo antedicho:



Podemos cambiar del *modo de presentación* al *modo de ejecución* a través del menú *View -> Presentation* o con el botón que se encuentra en la parte inferior del patcher y que vemos en la figura 5. Para editar cada modo, como ya explicamos, vamos a *View -> Edit* ó + e ó + clic en Macintosh y Ctrl. + e en Windows. Aclaremos que el modo de presentación nos permite mostrar sólo los objetos del patch que hayan sido incluidos en él, esto se realiza seleccionando el objeto y escogiendo la opción "add to presentation" en el menú "Object" o a través del shortcut + Shift + p (Mac) o Ctrl + Shift + p (Win). En la figura 6 vemos una de las formas en que podemos incluir a un objeto en el modo de presentación. Para ello, lo seleccionamos en el *modo de edición* del *modo de ejecución* y luego lo incluimos en el *modo presentación* a través de menú *Object -> Add to Presentation*. Una vez hecho esto veremos que el objeto se "ilumina" con un fondo de color rosado. Una vez realizado esto podemos cambiar al *modo de presentación* (tal como vimos en la figura 10) y allí, dentro del *modo de edición*, podemos cambiar la ubicación y el tamaño del objeto incluido:



Programación Orientada a Objetos (POO)

Se puede definir a la Programación Orientada a Objetos como una técnica o estilo de programación que utiliza objetos como bloque esencial de construcción.

Un patch de Max se construye a partir de la interconexión de bloques de programas (objetos) que realizan tareas específicas a partir de mensajes y atributos dados. Esta interconexión posibilita crear rutinas complejas combinando varias rutinas simples que pueden ser aisladas y comprendidas con facilidad. Dichos bloques realizan una serie de tareas que están ocultas al programador. En efecto, el usuario de Max no necesita saber cómo es que un determinado objeto accede a un puerto USB o al disco rígido. Simplemente sabe cual es su función y cómo puede interactuar con otros objetos. Cada objeto posee parámetros que pueden modificarse y hacer que su comportamiento sea diferente al de otro objeto del mismo nombre.

Objeto

En Max se llama objeto a una rutina invocada desde un *object box*. Cada objeto posee un nombre que le es propio y realiza una tarea específica; su comportamiento y propiedades varían de uno a otro. Por ejemplo, si escribimos dentro de un object box la palabra "print", estaremos invocando a una rutina que servirá para imprimir cualquier tipo de dato en el *Max Window*; mientras que si escribimos "sqrt" estaríamos implementando una operación de raíz cuadrada (figura 7).

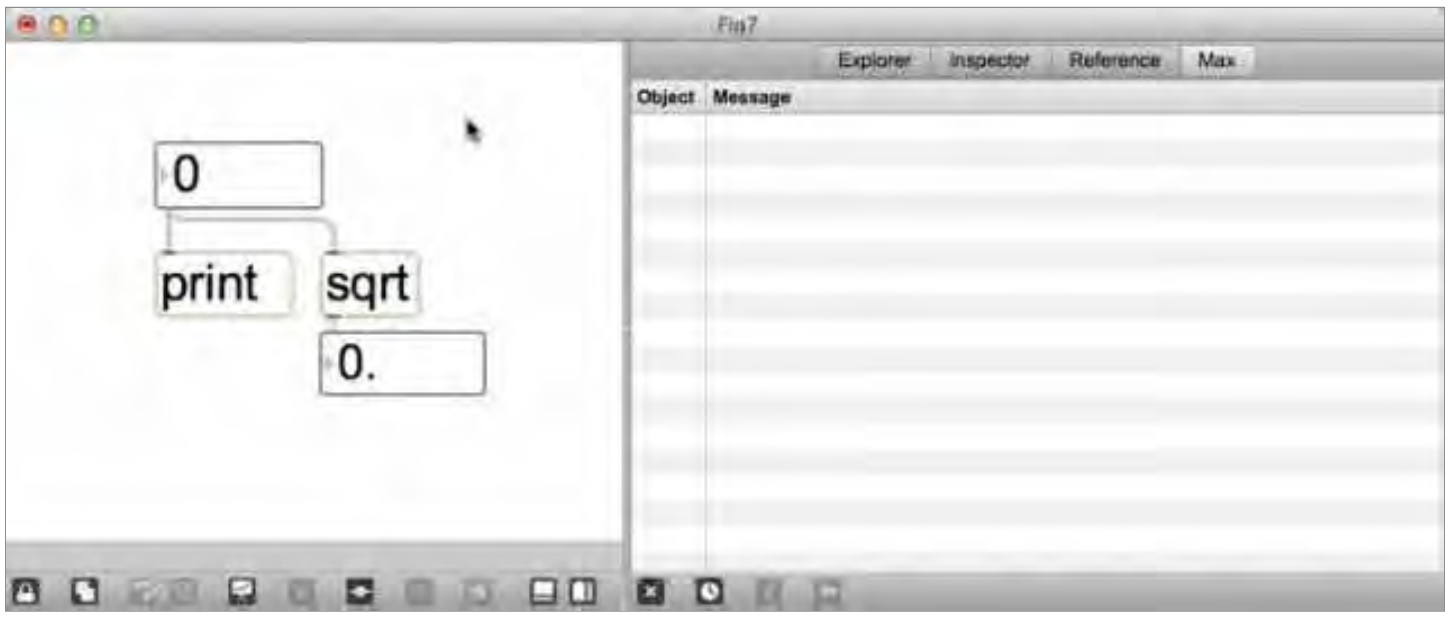


Figura 7

Existen una cantidad fija de objetos que se instalan automáticamente cuando instalamos Max en nuestra computadora, sin embargo es posible descargar muchos nuevos objetos desde la Web, realizados por diferentes programadores alrededor del mundo. Podemos acceder a ejemplos de estas colaboraciones a través del link *Get New Tools* que encontramos en la ventana *Getting Started*... en el menú *help*.

Los objetos se comunican entre sí a través de cables virtuales que se conectan desde los outlets (salidas) de unos, a los inlets (entradas) de otros (a partir de Max 6 es posible conectar desde un outlet a un inlet y viceversa). Es posible conectar varios objetos a uno o uno a varios (figura 8).

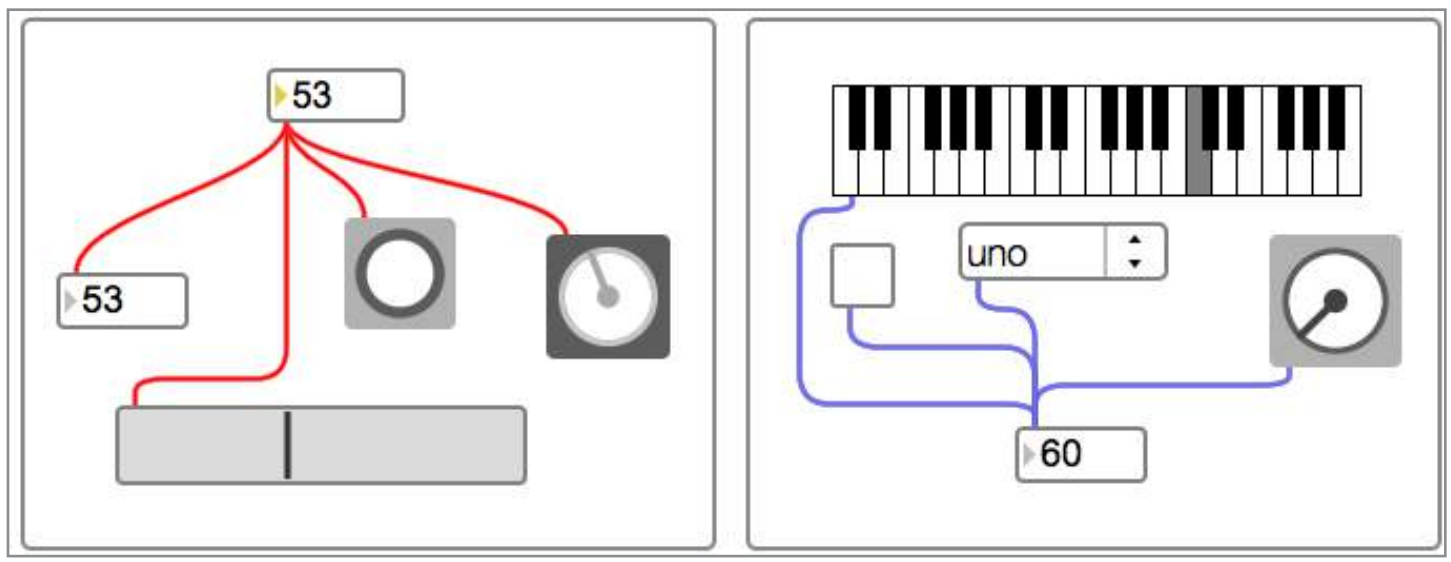


Figura 8

La mayoría de los objetos poseen inlets y outlets desde donde reciben y envían información. Existen algunos, como *print*, que sólo poseen un inlet y otros, como *receive*, que sólo poseen un outlet; existen objetos que tienen una cantidad fija de inlets y outlets (como *metro*), y otros una cantidad variable (como *gate*). Los vemos en la figura 9.

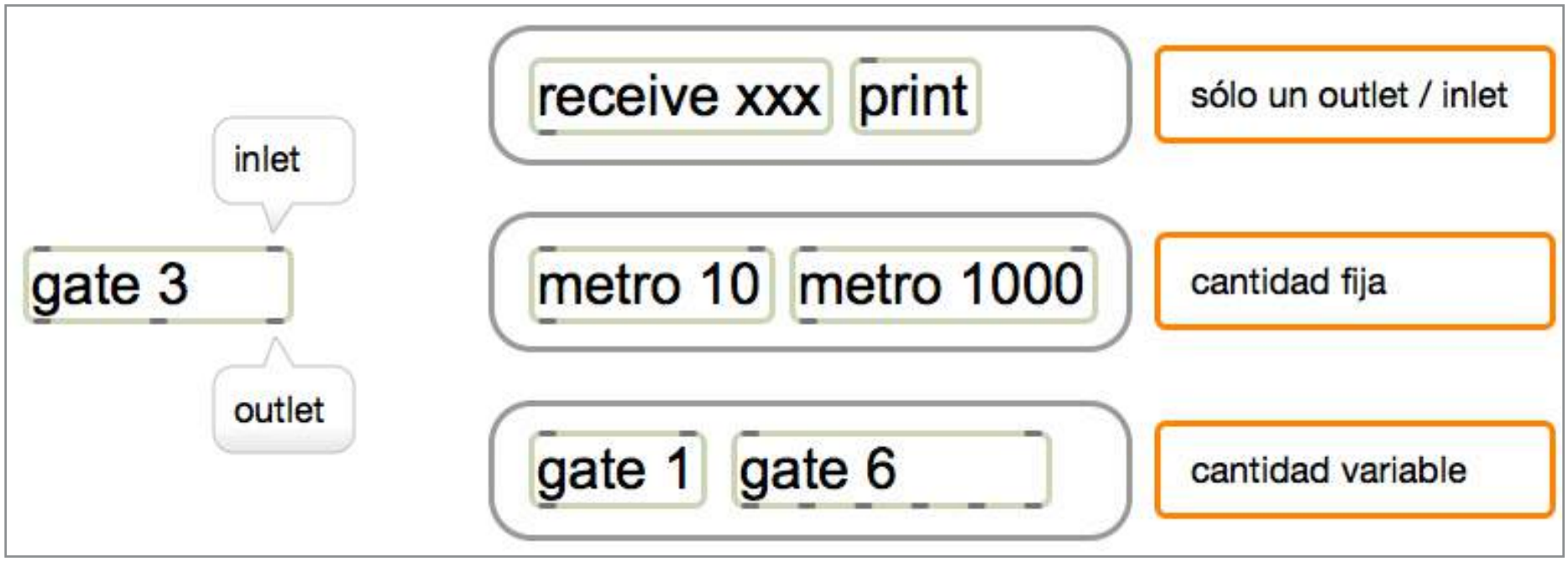


Figura 9

Object Box

Podemos decir que el *object box* es la herramienta principal de Max, ya que desde allí invocaremos programaciones con una funcionalidad propia (objetos). Éstas podrán interconectarse para crear otras más complejas.

Luego de colocar este objeto en el patcher, debemos escribir dentro de él lo que será el nombre que tiene asignada dicha rutina. Este nombre no es una palabra cualquiera sino el nombre exacto de un objeto existente dentro de Max. Si el objeto no existe (por ejemplo si escribimos el símbolo xx dentro de un *object box*) veremos en el *Max Window* la siguiente frase:

newobj: xx: No such object

En la figura 10 vemos tres ejemplos de objetos que realizan tareas muy diferentes entre sí: *cycle~*, *noteout* y *matrix~*.

Figura 10

Cuando colocamos un nuevo objeto en nuestro patcher y comenzamos a tipear una determinada palabra dentro de él, aparecerá una lista con todos los objetos que utilizan las letras ingresadas:

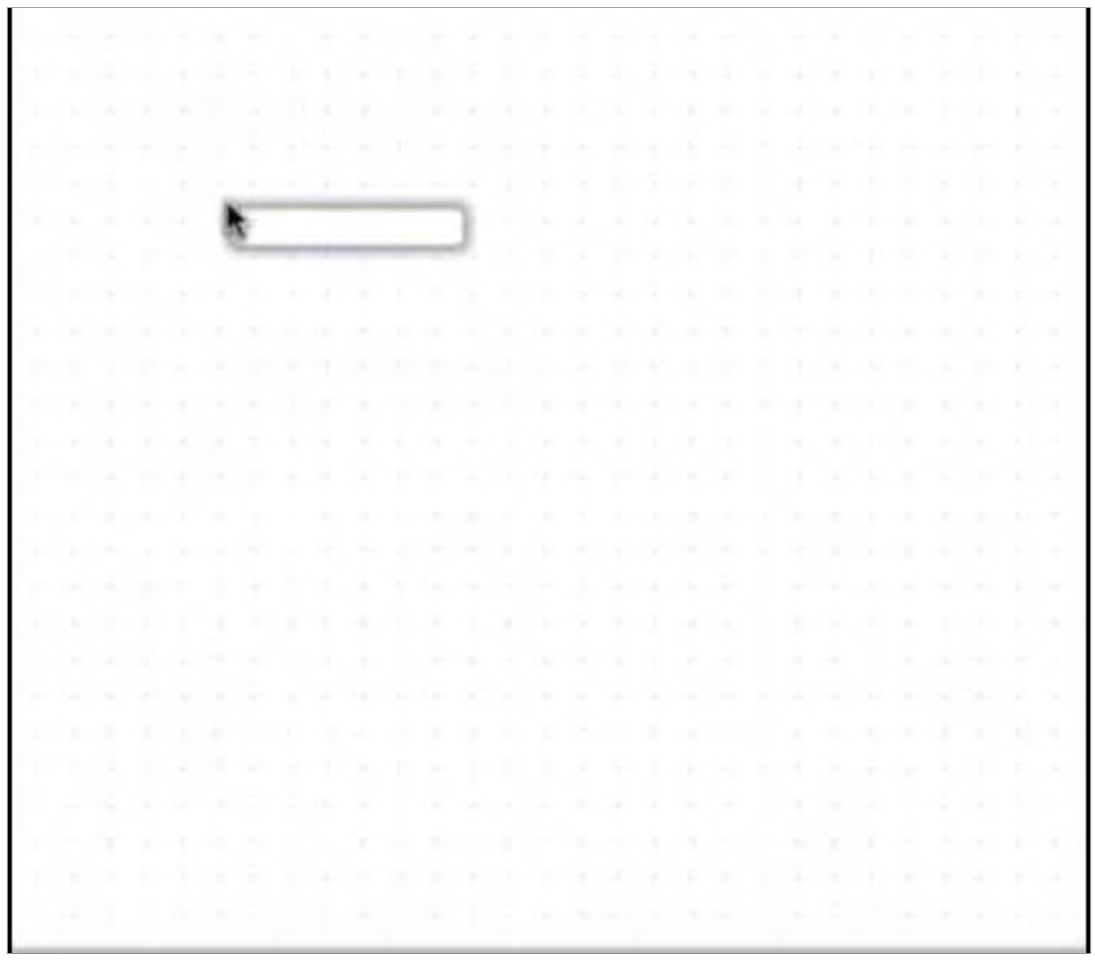


Figura 11

Argumentos

En ciertas ocasiones, a continuación del nombre del objeto, escribimos alguna palabra o número (o una combinación de ambos) que denominamos argumentos. Éstos sirven para inicializar† un determinado parámetro o para especificar cierta característica que el objeto poseerá (como por ejemplo el número de inlets y/o outlets). En algunos casos los argumentos son opcionales (como en el caso de *gate*) y en otros obligatorios (como en el caso de *adstatus*). En la figura 12 podemos ver cómo, en el caso de *gate*, si no colocamos ningún argumento después del nombre, el objeto se inicializa con un solo outlet, mientras que si colocamos como argumento el número 5 lo hace con cinco outlets. Por otro lado en el caso de *adstatus* vemos que al no colocar ningún argumento, Max envía un mensaje de error en el *Max Window* donde nos indica que el objeto debe poseer obligatoriamente un argumento.

Los argumentos deben escribirse en un orden específico, ya que la posición en la que se encuentran dictamina el parámetro al cual se refieren. Por ejemplo en el caso de *gate*, si colocamos dos valores como argumento, el primero será la cantidad de salidas y el segundo la salida que por defecto estará abierta.

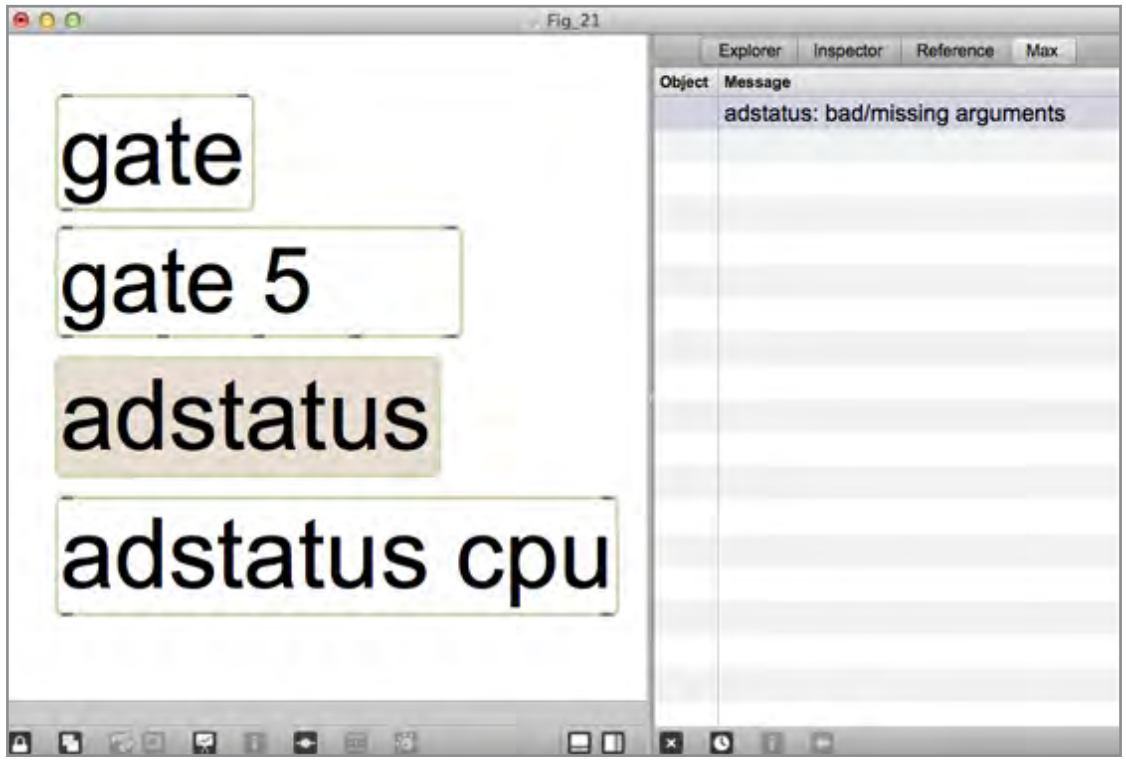


Figura 12

† Es decir que utilizará ese argumento desde el inicio.

Cuando utilizamos argumentos dentro de un objeto debemos tomar en cuenta que dichos valores sólo son efectivos mientras no ingrese una información nueva que modifique esos datos. De ser así el valor de inicialización se perderá hasta que abramos el patcher nuevamente. Por ejemplo, si colocamos un operador matemático de suma (+), con un valor de inicialización de 10, el resultado será la suma del valor ingresado a través del primer inlet más 10, pero si luego ingresamos un 20 por el segundo inlet de dicho operador, el resultado final será el valor ingresado más 20 y el 10 queda obsoleto hasta que abramos el patch nuevamente.

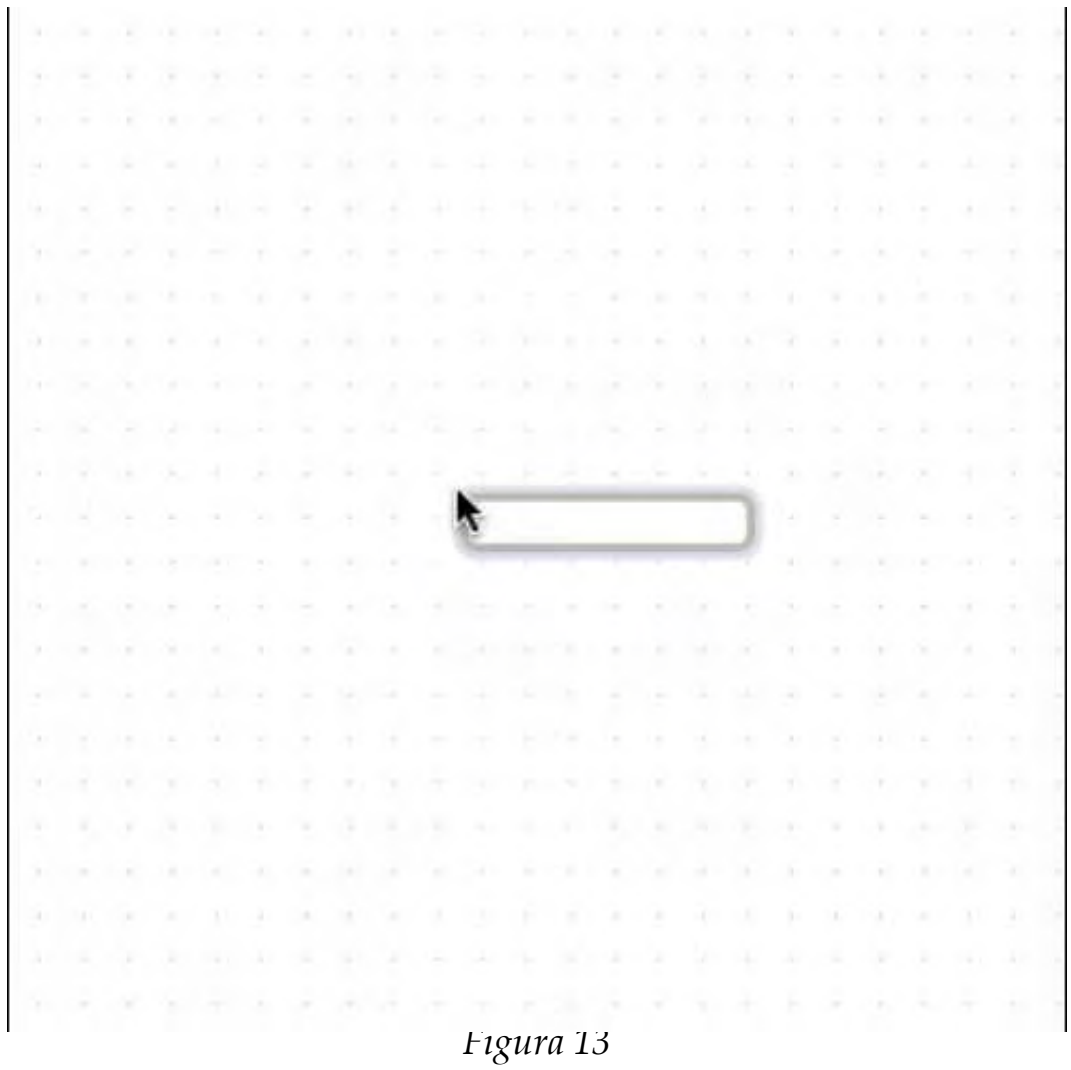


Figura 13

Mensajes

Como mencionamos anteriormente, los objetos reciben y envían mensajes a través de los inlets y outlets. Estos mensajes pueden ser números enteros (*int*), números flotantes o decimales (*float*), listas de números (*list*), palabras (*symbol*), *bangs* o una combinación de todo esto (*any message*). Estos mensajes pueden ser enviados a un objeto a través de una caja de mensaje (*message box*).

No deben confundirse los *message box* con los *object box*. El *object box* es la herramienta más importante de Max y es desde donde vamos a invocar a una rutina determinada; mientras que el *message box* sirve para enviar algún tipo de mensaje. Para diferenciar fácilmente uno del otro podemos decir que, como vemos en la figura 14, el *object box* tiene un borde o línea de contorno y es blanco en su interior, mientras que el *message box* no tiene una línea de contorno y es gris.

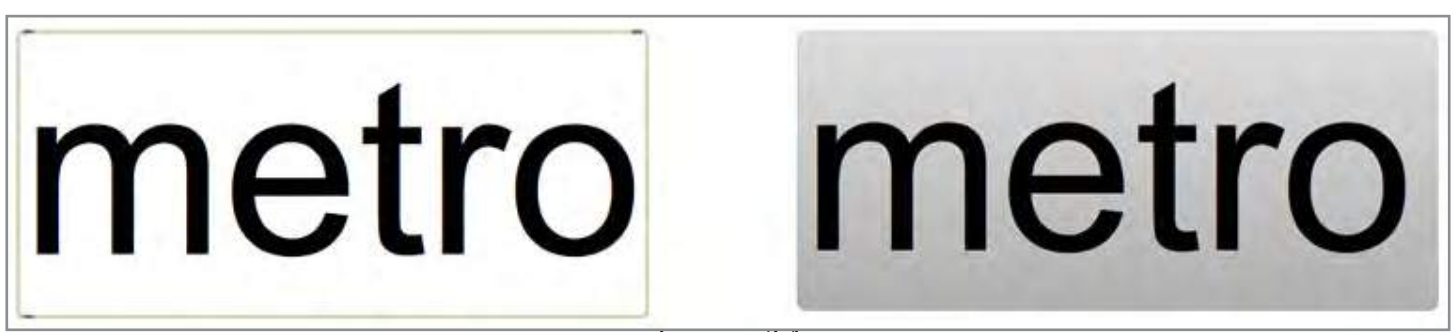


Figura 14

En la figura superior vemos, del lado izquierdo, un objeto llamado *metro* que realiza un tipo de función específica, mientras que el objeto de la derecha es un mensaje "metro", enviado a través de un *message box*, que podrá o no incidir en el comportamiento de alguna rutina. Por último diremos que en el *modo de ejecución*, el *message box* se convierte en un botón virtual, es decir, cuando hacemos clic sobre él, la caja "se anima" como si se hundiera (apretamos un botón) y envía el mensaje que se encuentra escrito dentro de él.

Tipos de mensajes

A continuación detallamos los diferentes tipos de mensajes que Max envía y recibe:

float: números con coma flotante (por ejemplo 0.23).

int: número entero, sin decimales (por ejemplo 127).

list: lista de dos o más números flotantes o enteros separados por espacio (por ejemplo 100 3.23 0.0002 1834).

bang: este es un tipo especial de mensaje que "le dice" al objeto: "haz lo que haces, sea lo que sea". Por ejemplo si enviamos el mensaje bang al objeto *random*, éste generará un número aleatorio dentro de un rango dado, mientras que si el bang es enviado a un objeto *counter*, incrementaremos su cuenta un paso.

symbol: es una palabra. Se utiliza para modificar algún parámetro preestablecido del objeto o para que éste realice una determinada acción. Por ejemplo el objeto *capture*, entre muchos otros, acepta la palabra "open". Este mensaje abrirá una ventana desde donde podremos ver ciertos datos. Cada objeto acepta mensajes específicos, por lo tanto si se le envía una palabra que no sea aceptada por éste, veremos en el *Max Window* un mensaje de error.

any message: es posible enviar mensajes combinando palabras y valores (por lo general un parámetro específico junto a uno o más números). Si, por ejemplo, le enviamos al objeto gráfico *panel* el mensaje "brgb 0 255 0" éste cambiará su color de fondo a verde , mientras que si le enviamos el mensaje "brgb 255 0 0" cambiará a rojo.

Existen objetos que aceptan todos estos tipos de mensajes (como *expr*) y otros que sólo aceptan algunos (como el objeto *int*).

En la figura 15 podemos ver cómo el objeto *Keyboard Slider* o *kslider* acepta cinco tipos de mensajes diferentes. Si le enviamos un int, mostrará la nota MIDI a la que ese entero se refiere (siendo su key velocity 64). Si le enviamos un list de dos valores, tomará al primero como la nota MIDI y al segundo como su veloci-

ty. Si, en cambio, enviamos un symbol “chord” más una lista, considerará cada grupo de dos valores de esa lista como un grupo de nota + velocity. En el caso de enviarle un symbol “color” acompañado de un entero podemos cambiar el color de las teclas activas (en este caso “color 0” mostrará las teclas en gris). Por último el symbol “clear”, sin agregado de ningún otro mensaje, borrará las teclas activas.

También mencionemos que, cuando enviamos un mensaje con diferentes valores o palabras separadas por una coma, éstos saldrán de manera iterada (es decir uno a continuación del otro).

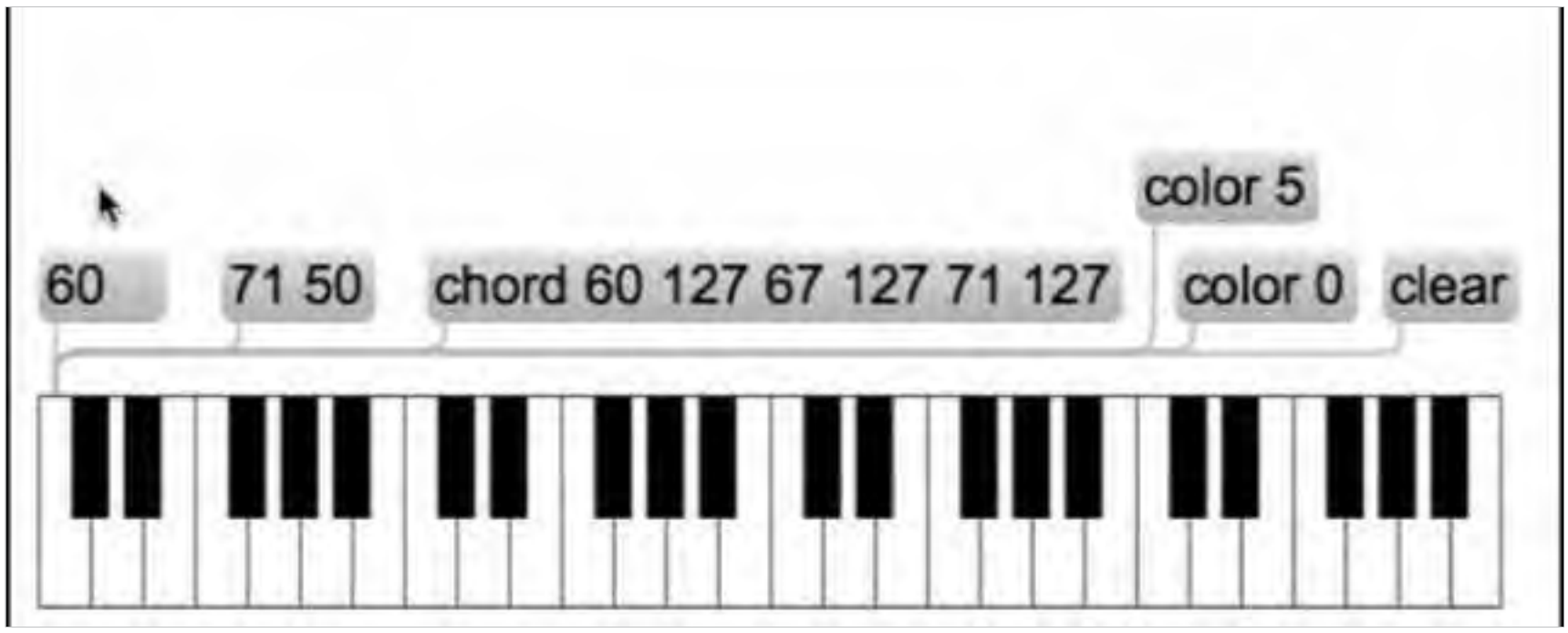


Figura 15

Variables

Es posible enviarle a un objeto, un mensaje acompañado de una o más variables declaradas a través del signo $ más el número de variable (por ejemplo $1). En la figura 16 vemos en el *Max Window* que, al colocar dicho signo a continuación de un texto cualquiera (en este caso la palabra “color”), el texto se mantiene fijo mientras que el número que ingresa por el primer inlet del *message box* es la parte que varía (Como vemos, declarada con el signo $1). Por otro lado, como el objeto *button* puede recibir el mensaje “color” acompañado de un ente-

ro, al colocar ese entero como variable podemos modificar el color de *button* en tiempo real.

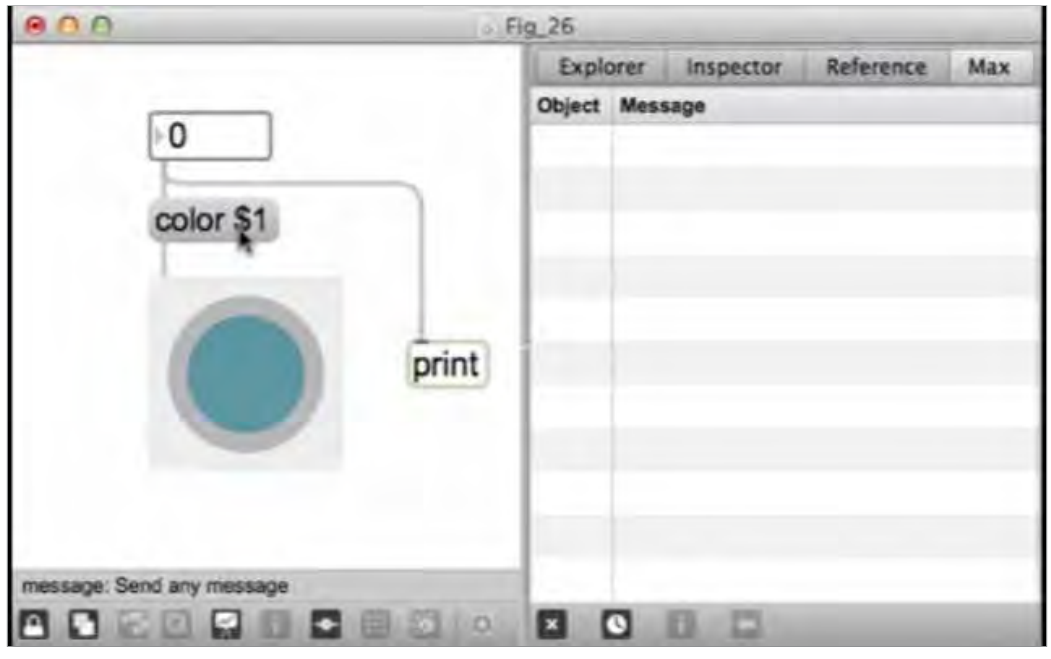


Figura 16

Atributos

A partir de la versión 4.5, algunos objetos como *mxj*, *pattr*, *pattrstorage*, *pattrhub* y *autopattr* comenzaron a utilizar atributos. A partir de Max 5, esta función fue extendida a muchos más objetos. Como ya vimos, en el caso de los argumentos, Max "sabe" a qué parámetro se refiere por la posición en la que se encuentra escrito; esto conlleva el problema de que, si queremos inicializar un parámetro que está definido como el argumento que se encuentra en el segundo lugar, debemos inicializar a la fuerza también el que se encuentra en primero; ahora bien, cuando usamos atributos en lugar de argumentos no hay inconveniente en cambiar el orden de aparición de los parámetros ya que le indicaremos al objeto en cuestión cuál atributo vamos a inicializar anteponiendo, a la palabra que designa el parámetro, el símbolo @. Vemos un ejemplo en la figura 17

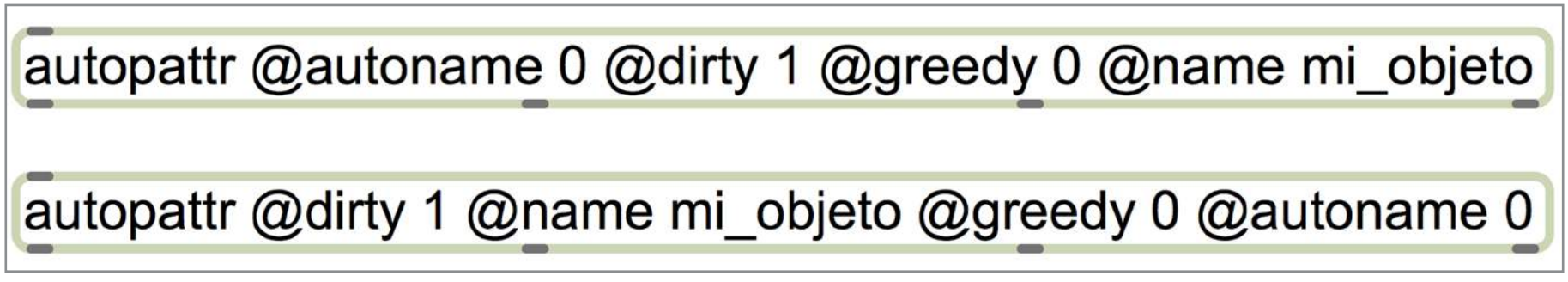


Figura 17

Como podemos observar en la figura 17, el atributo “autoname” tiene delante el símbolo @, esto le indica al objeto que la palabra adherida al mismo será el nombre del atributo y el valor siguiente será un argumento de ese atributo. En este caso *@autoname 0* significa que el atributo autoname tiene un argumento 0 (en este momento no es importante saber qué significa esta orden para el objeto *autopattr*). Es por esta razón que no importa el orden en que se coloque el atributo, ya que Max entiende esa orden por su nombre y no por su posición. En el ejemplo de la figura 18 vemos que el objeto *metro* se “enciende” cuando abrimos el patch sin necesidad de activarlo manualmente.

Existen una serie de atributos que son compartidos por todos los objetos. Éstos pueden ser configurados a través del *inspector* o enviando un mensaje antecedido por la palabra “sendbox” y precedido por el o los datos requeridos. La lista completa de estos atributos comunes la veremos a continuación cuando expliquemos el *inspector (Figura 19)*.

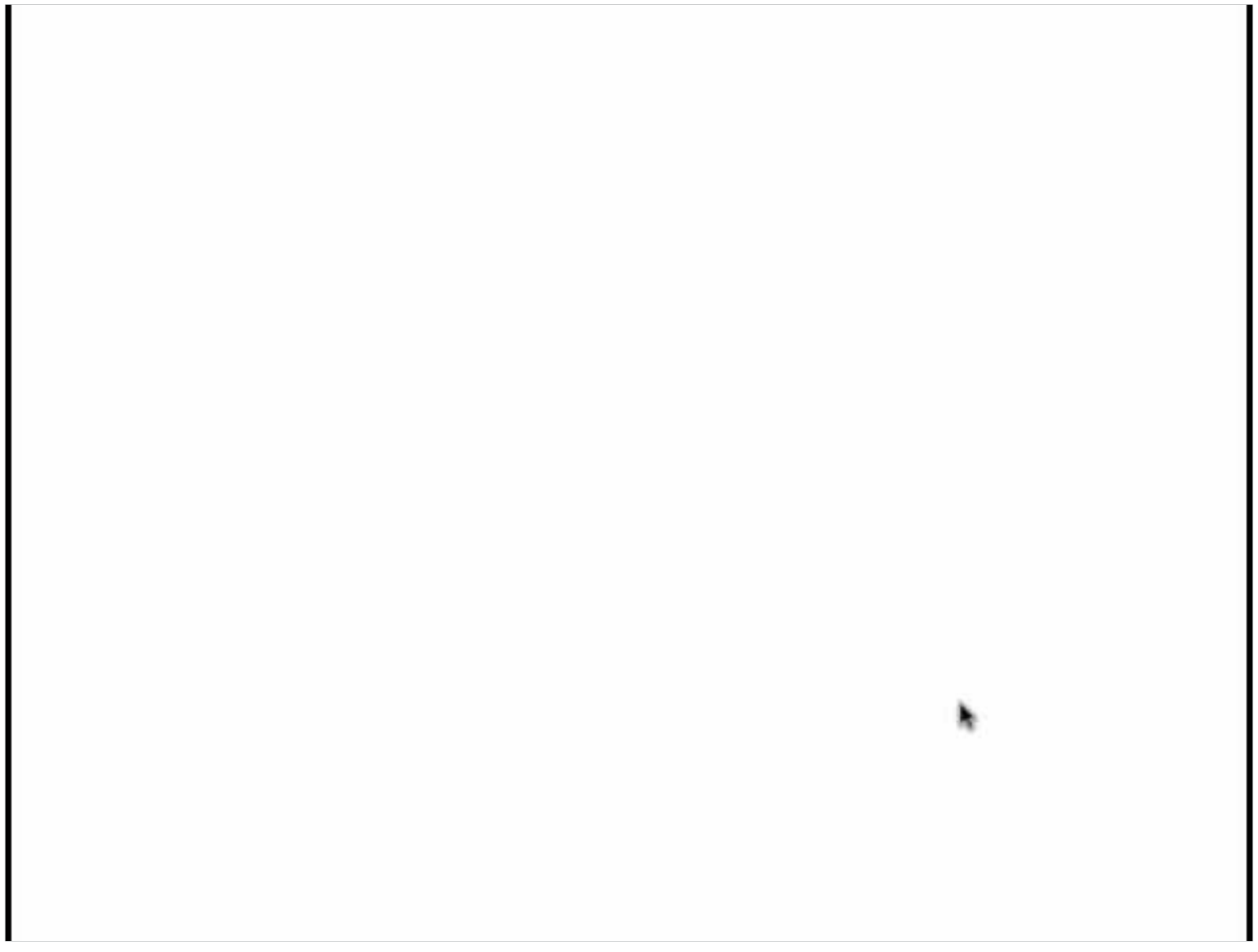


Figura 18

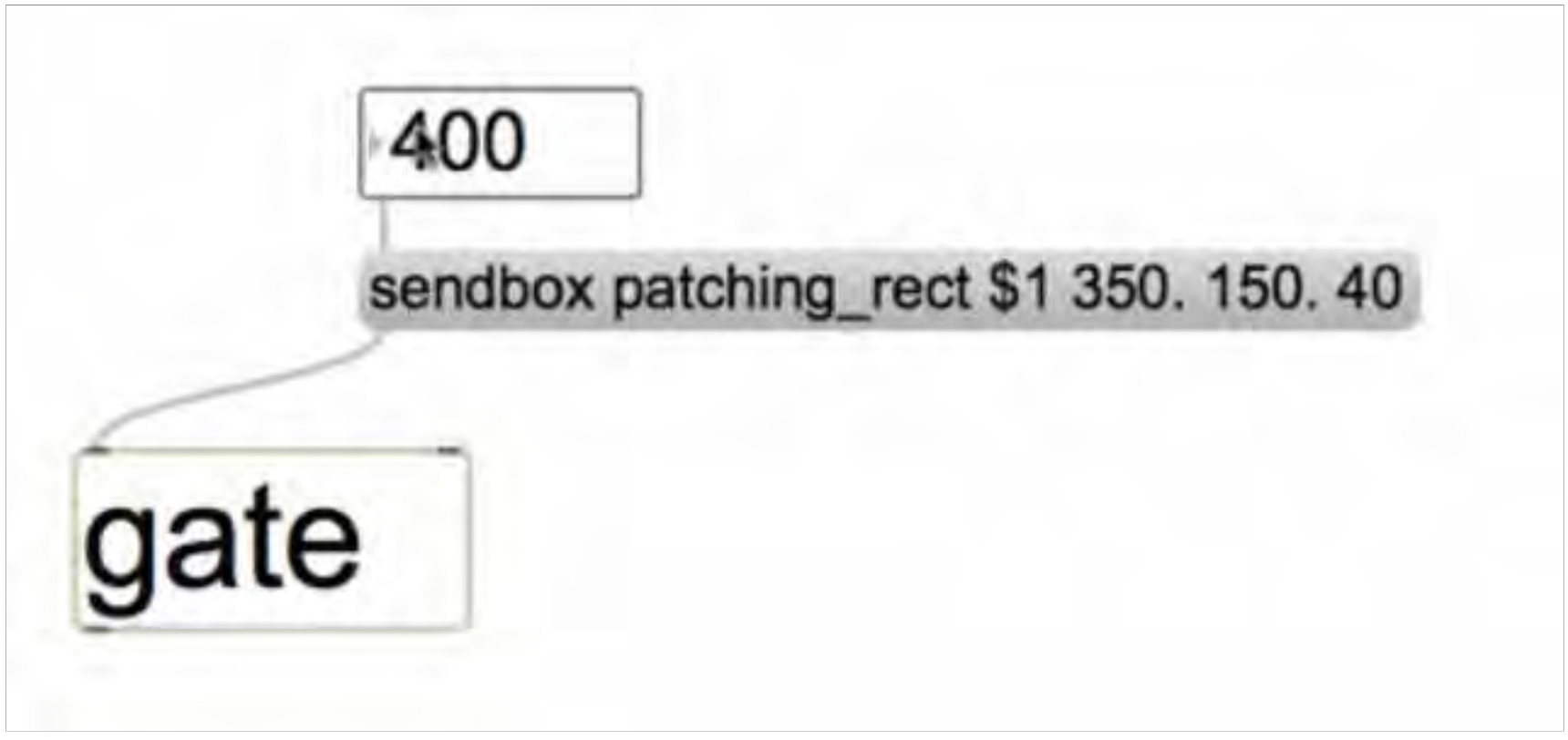


Figura 19

En este caso el atributo común “patching_rect” será capaz de configurar el tamaño y la ubicación del objeto en el patch, por lo que, al enviar este atributo antecedido por la palabra “sendbox” y precedido por una variable ($1) y tres flotantes (que serán la ubicación horizontal, la ubicación vertical, ancho y alto del objeto respectivamente), podremos controlar en tiempo real la ubicación horizontal del objeto. Así, en este ejemplo, podemos mover al objeto *gate* a través del patch desde el número flotante que ingresa al *message box*.

Inspector

Existen dos tipos de ventanas de inspector en Max: el *Object Inspector* que permite configurar parámetros y preferencias de los objetos y el *Patcher Inspector* que hace lo mismo pero aplicado a los patcher. Veamos cada uno:

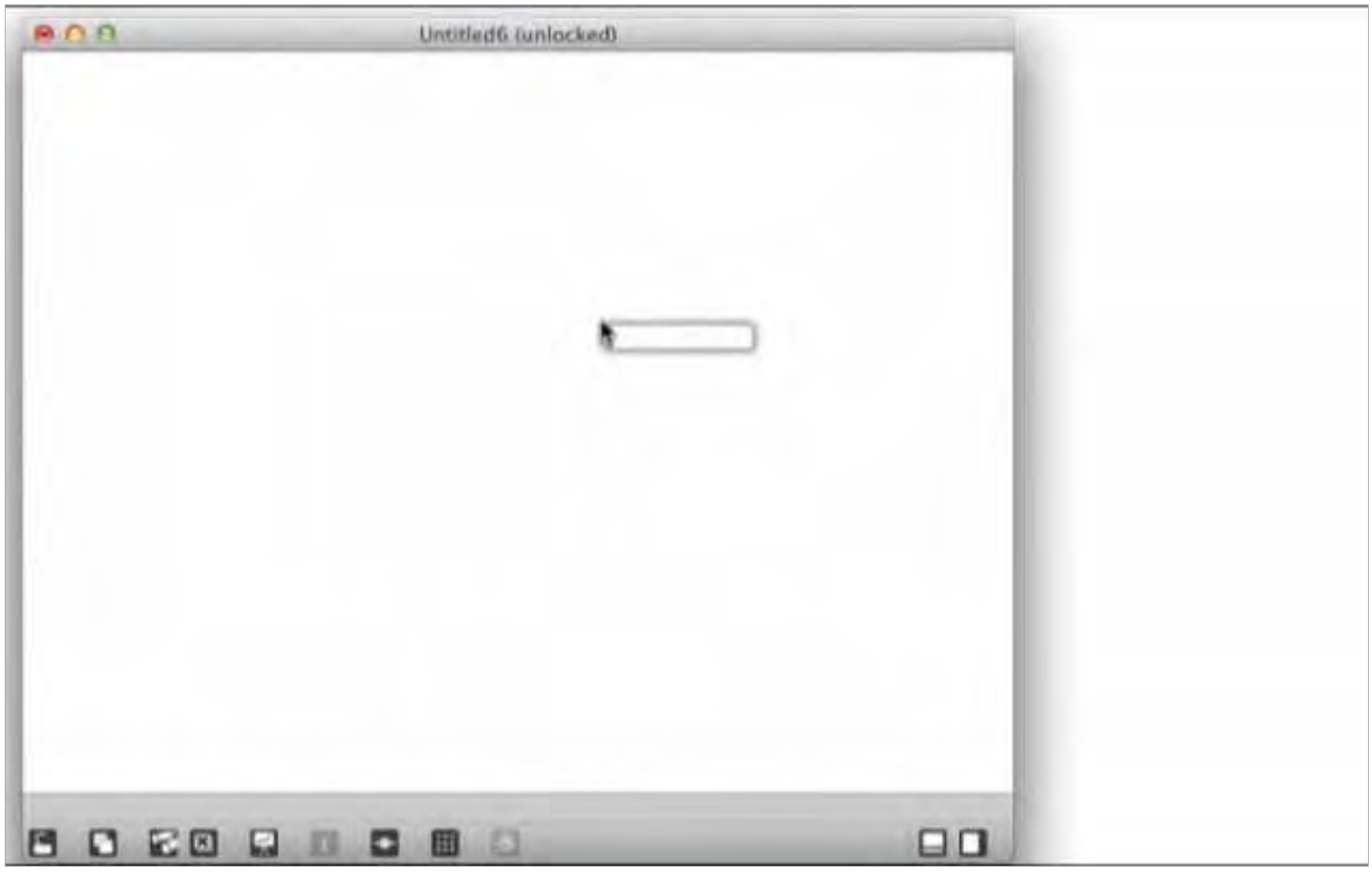


Figura 20

Object Inspector: existen cuatro maneras de abrir la ventana de inspector de un objeto dado. La primera (Fig. 20) cuando hacemos clic sobre la porción azul de la rueda que aparece cuando posamos el puntero del mouse en el lado izquierdo de un objeto. Allí nos aparecerá un menú desde donde podremos escoger el *inspector*. La segunda (Fig. 21), cuando seleccionamos uno o más objetos y luego hacemos clic en el botón de *inspector* que está en la parte inferior del patch. La tercera (Fig. 22), cuando seleccionamos uno o más objetos y escogemos el submenú *inspector* que se encuentra dentro del menú *Object*. Por último, si dejamos abierta la ventana de *inspector* que se encuentra en el Sidebar del patcher, al seleccionar diferentes objetos dicha ventana mostrará los parámetros correspondientes de esos objetos.

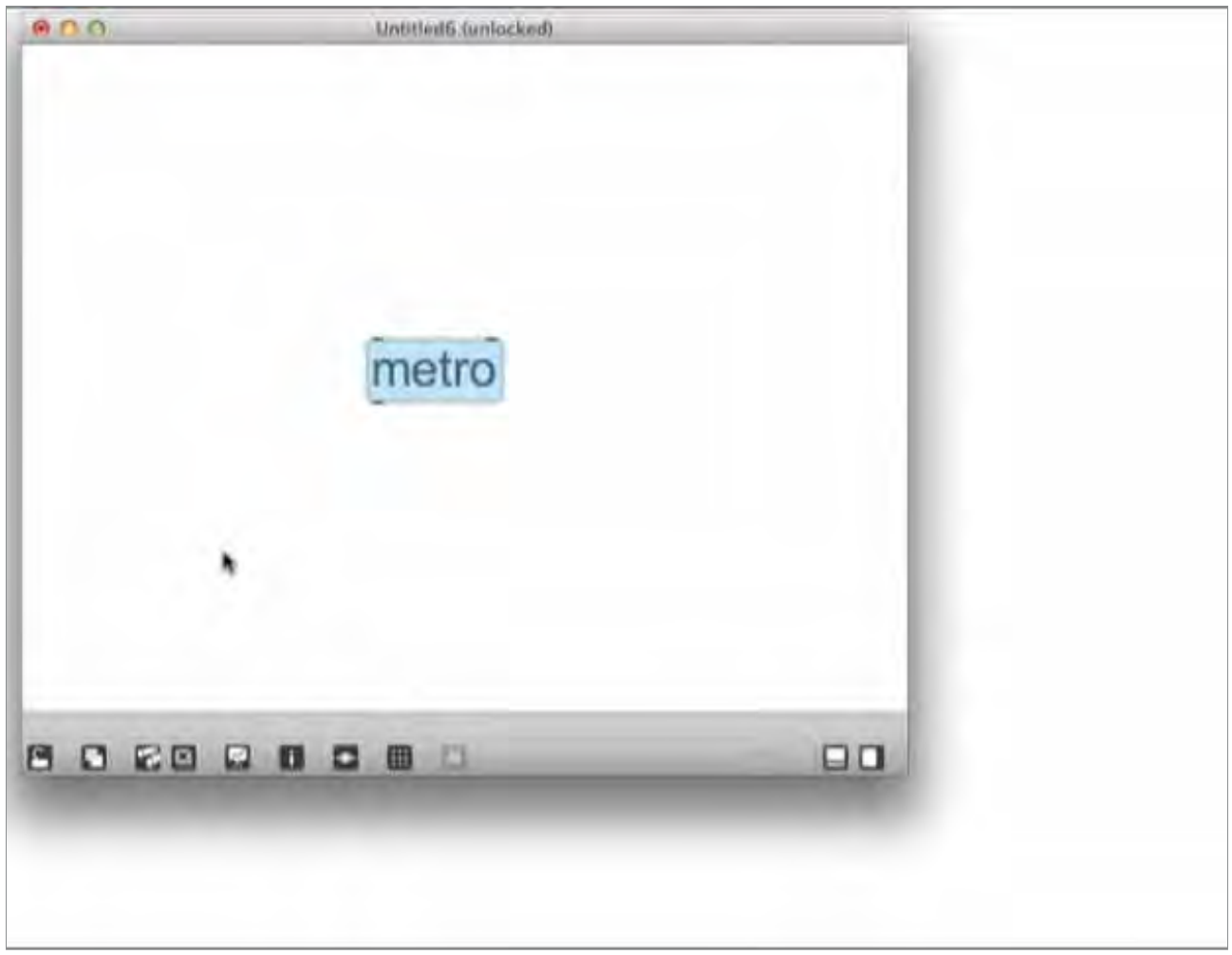


Figura 21

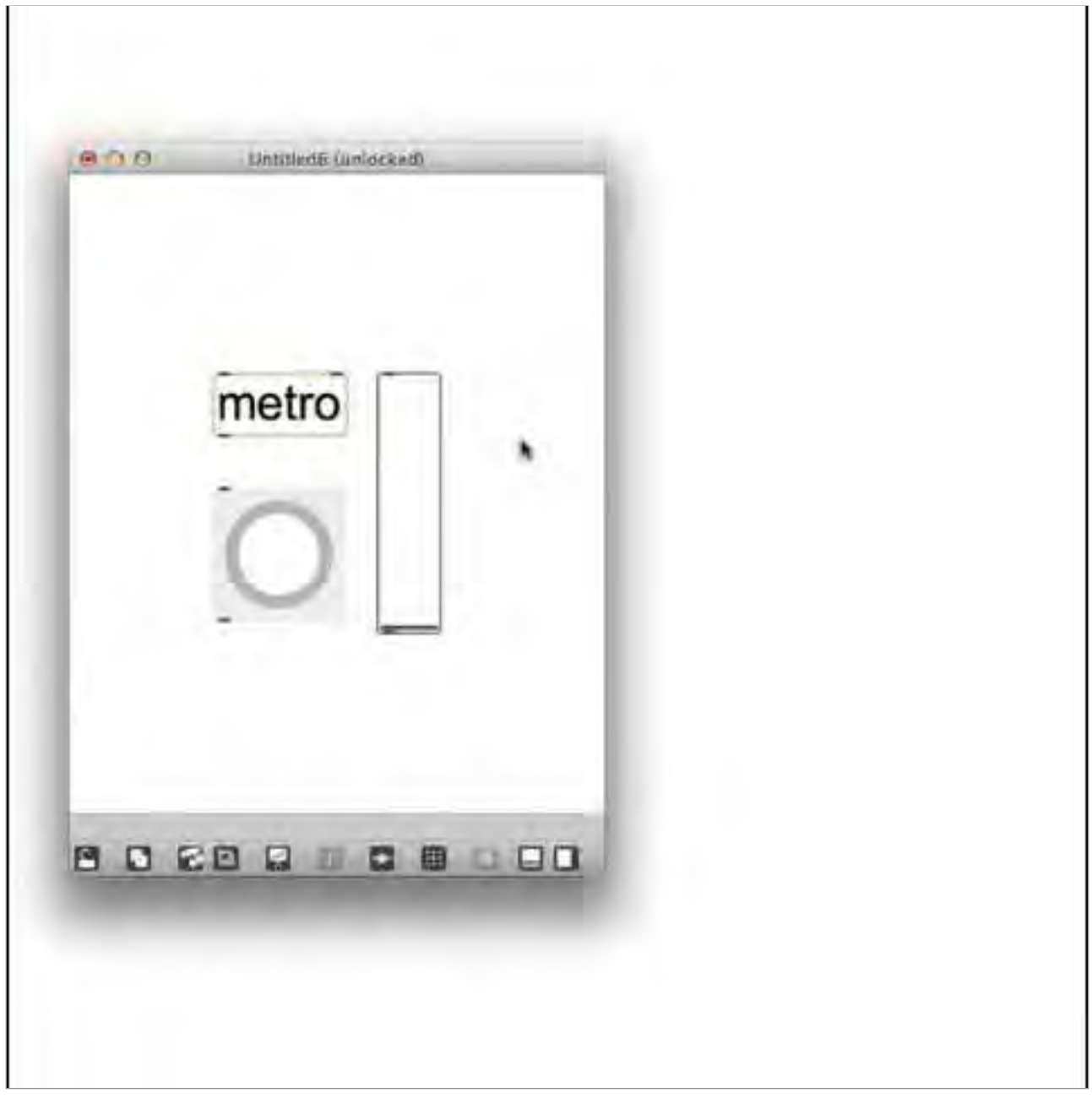


Figura 22

Como ya mencionamos, hay atributos comunes a todos los objetos (*Common Box Attributes*) y otros que son propios de cada uno. Veamos a continuación los que son comunes a todos:

hidden: oculta un objeto cuando el patcher no está en el *modo de edición*.

background: incluye o no un objeto en el patcher background. De esta manera podemos anclar la posición de un objeto para evitar que pueda moverse. Si seleccionamos el submenú *Lock Background* dentro del menú *View*, los objetos que tengan activa la casilla background no podrán moverse.

presentation: incluye al objeto en el *modo de presentación*.

patching_rect: configura la posición y tamaño del objeto en el patcher. Este atributo utiliza cuatro números flotantes: posición horizontal, posición vertical, largo del objeto y alto.

presentation_rect: lo mismo que el anterior pero para el modo presentación.

ignoreclick: hace que el objeto no pueda interactuar con el mouse cuando se encuentra en los modos edición y presentación.

color: configura el color del contorno del objeto.

textcolor: lo mismo que lo anterior pero para el texto dentro del objeto.

fontname: configura la tipografía que utilizará el texto dentro del objeto.

fontsize: configura el tamaño de la tipografía utilizada.

fontface: selecciona el estilo de tipografía a utilizar (plain, bold, italic, bold italic).

annotation: configura el texto que se mostrará en el *Clue Window* cuando pasemos el puntero del mouse por sobre el objeto. El *Clue Window* (o ventana de pista) es una ventana que aparece con la versión 5 de Max y que nos da, como su nombre lo indica, una pista acerca de la función que cumple el objeto sobre el cual coloquemos el puntero del mouse. Si ingresamos algún texto específico a través del atributo "annotation" entonces mostrará dicho texto, si no, por defecto, nos dirá brevemente la función que cumple (figura 23). El *Clue Window* podemos abrirlo desde el menú *Window -> Clue Window*.

Figura 23

hint: configura el texto que mostrará el objeto cuando coloquemos el puntero del mouse sobre él estando en el modo de ejecución o de presentación. Este texto aparece en un "pop-up" como el que se muestra a continuación.

Figura 24

varname: permite asignarle al objeto, un nombre que se utilizará en el contexto de un "script". Esto será de gran utilidad cuando utilicemos objetos como *pattr*, *thispatcher* y *js*.

Con respecto al *Patcher Inspector*, es posible acceder a él con el botón derecho del mouse haciendo clic sobre cualquier parte vacía del patcher o, sobre un patcher abierto, sin ningún objeto seleccionado, haciendo Cmd + Shift + i en Macintosh u Option + Shift + i en Windows. Los parámetros que pueden configurarse desde allí afectan al comportamiento e interfaz del patcher. A continuación se detallan:

Background Color: permite configurar el color de fondo del patcher, cuando éste se encuentre en el modo de ejecución.

Text Editing Frame Color: permite cambiar el color del borde de un objeto cuando éste se encuentra en el estado en el cual es posible escribir o editar el nombre que contendrá.

Unlocked Background Color: permite configurar el color de fondo del patcher cuando éste se encuentre en el modo de edición.

Box Animate Time: especifica el tiempo, en milisegundos, que utilizan los objetos para ir de la configuración del modo de ejecución a la del modo de presentación.

Grid Size: configura el tamaño de la grilla que utilizará el patch. Ésta se puede activar desde el menú *View -> Grid* (para que la muestre o no) y *View -> Snap to Grid* (que permite mover los objetos dentro del patch pero sólo pegados a la grilla).

Save Default-Valued Object Attributes: si seleccionamos esta casilla, el patch no se verá afectado por las configuraciones que se hayan realizado desde el

menú *Option -> Object Defaults.*

Default Font Name: configura la fuente que utilizarán los objetos dentro del patch.

Default Font Size: configura el tamaño de fuente que utilizarán los objetos dentro del patch.

Default Font Style: configura el estilo de la tipografía (bold, itálica, etc.) utilizada en el patch.

Help

Cada objeto de Max posee su propia ayuda que consiste en un patch de ejemplo, totalmente funcional, realizado con el objeto en cuestión y que sirve para aclarar su funcionamiento. Esta ayuda se puede abrir seleccionando "O*pen ... Help*" en el menú que aparece haciendo clic con el botón derecho del mouse sobre cada objeto o con Option + Clic (en Mac) sobre el mismo objeto. También podemos abrirlo desde la rueda que aparece cuando ponemos el puntero del mouse en el borde izquierdo de un objeto (en *modo de edición*). Allí escogemos la porción azul de la rueda y allí veremos un menú desde donde podremos escoger el help.

Además, es posible abrir un referencia del objeto desde el patch de ayuda. Para ello hacemos clic sobre la pestaña "?", y allí en el ítem que dice *Open Reference*. Otra forma de ayuda para un objeto determinado la podemos obtener colocándonos con el puntero del mouse sobre un inlet u outlet. Allí veremos un mensaje (encerrado en un globo) que nos indicará brevemente la función que dicha entrada o salida cumple. Por último, si hacemos clic con el botón izquierdo del mouse sobre un inlet, accederemos a una ventana desde donde es posible abrir la ayuda para dicho objeto, o abrir la ventana de referencia y conocer todos

los mensajes y atributos que el objeto en cuestión acepta. Veamos un ejemplo de todo esto en la Figura 25.

Orden de los mensajes

En todos los lenguajes de programación, como pueden ser el C o JavaScript, los comandos se ejecutan en un orden específico, por ejemplo de arriba hacia abajo. En Max, en cambio, el orden de ejecución es de derecha a izquierda, es decir, que en un patch como el que muestra la Fig. 26 izquierda, el número 100 llegará primero al *number* C , luego al B y por último al A . Otro ejemplo lo vemos en la Fig. 26 derecha, donde estamos enviando al mismo tiempo cuatro mensajes que se imprimirán en el *Max Window* (a través del objeto *print*). El orden de aparición esperado es el que formaría la frase "mi nombre es Francisco", sin embargo vemos en el *Max Window*: "Francisco es nombre mi", ya que "Francisco" está en primer lugar contando de derecha a izquierda y "mi" está último. Tomar en consideración esta particularidad que tiene Max, será de gran importancia en ciertas programaciones que veremos más adelante.

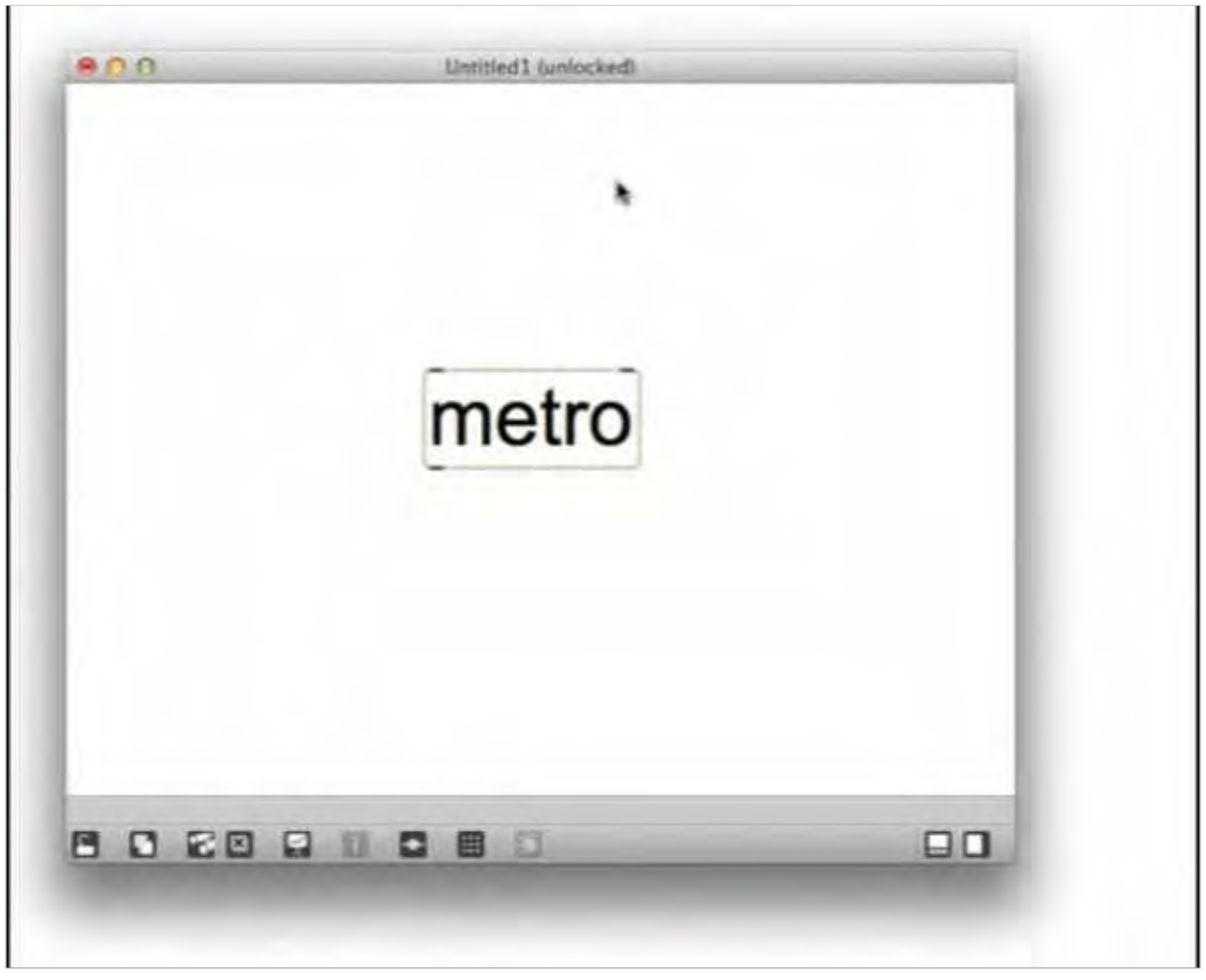


Figura 25

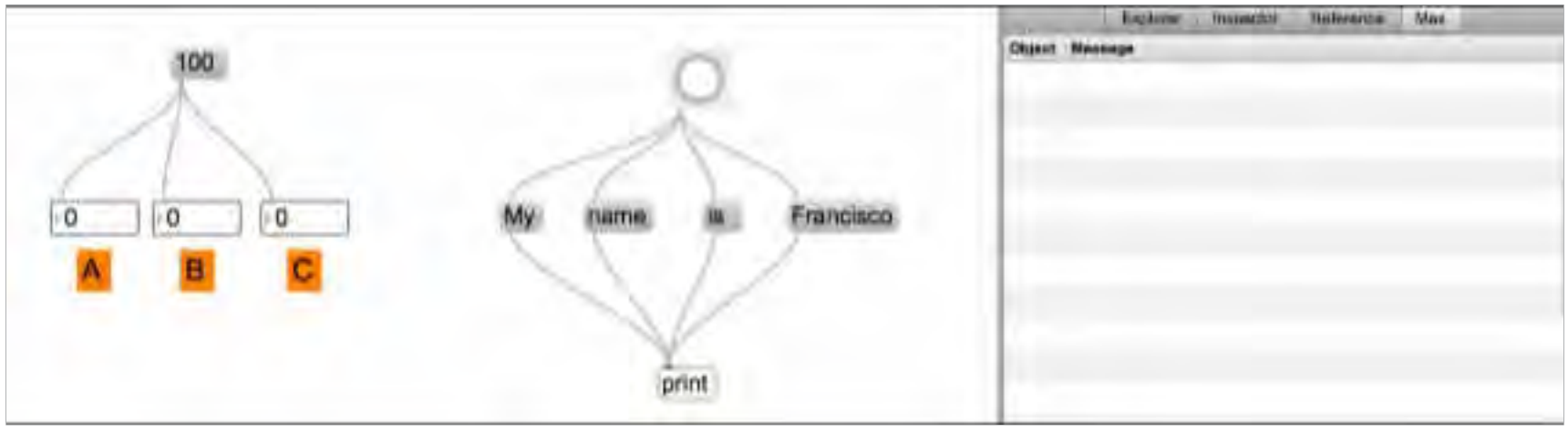


Figura 26

File Preferences

Cuando deseamos acceder a un archivo externo dentro de Max, por ejemplo un archivo de audio, video, un subpatch, etc... debemos indicarle a Max la ruta donde se encuentra dicho archivo. Esto puede realizarse de forma automática a través de dos métodos diferentes. El primero es indicarle la ruta a partir del submenú *File Preferences* que se encuentra dentro del menú *Options*. Allí será posible agregar la ruta para acceder a una carpeta (y a sus sub-carpetas) donde Max buscará cuando convocamos a un archivo determinado. En el siguiente ejemplo vemos cómo, al querer convocar a una imagen llamada "dogs.jpg" dentro del objeto *fpic* (a través del mensaje "pict dogs.jpg"), nada sucede (ya que Max no encuentra el archivo indicado en el mensaje). Luego cuando asignamos desde *File Preferences* la ruta donde se encuentra dicha imagen entonces la foto aparece inmediatamente al enviar el mensaje.

El segundo método que podemos utilizar para que Max encuentre un archivo es colocarlo en la misma carpeta donde se encuentra el patch desde donde queremos convocar a dicho archivo. Es importante destacar que deben encontrarse ambos al mismo nivel, es decir que no debe estar uno de ellos dentro de una sub-carpeta. En el ejemplo de la figura 28 vemos que, a pesar de que la ruta donde se encuentra la fotografía no se está configurada en el *File Preferences*, la imagen aparece de todos modos ya que se encuentra en la misma carpeta donde está el patch que utilizamos para convocarla.

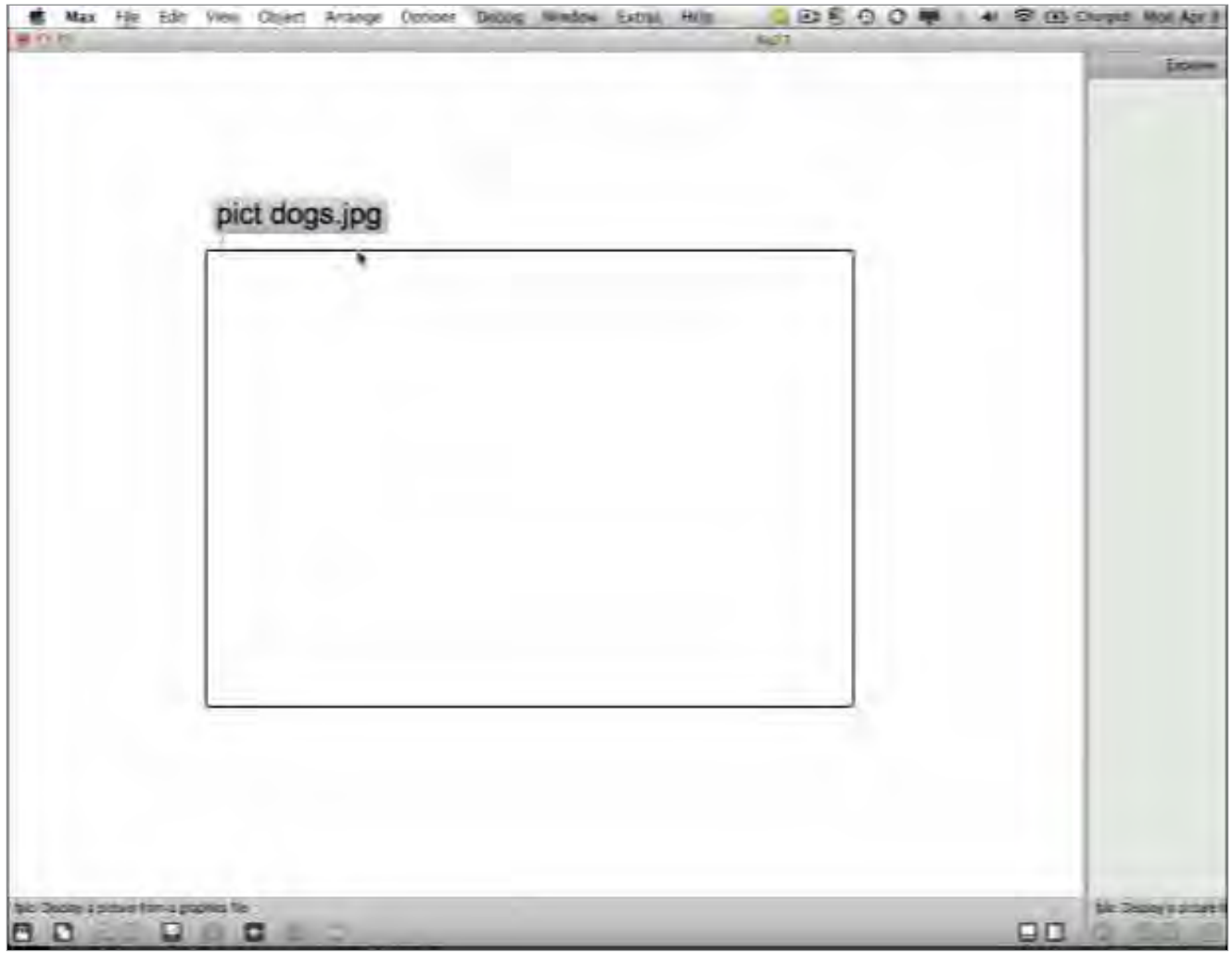


Figura 27

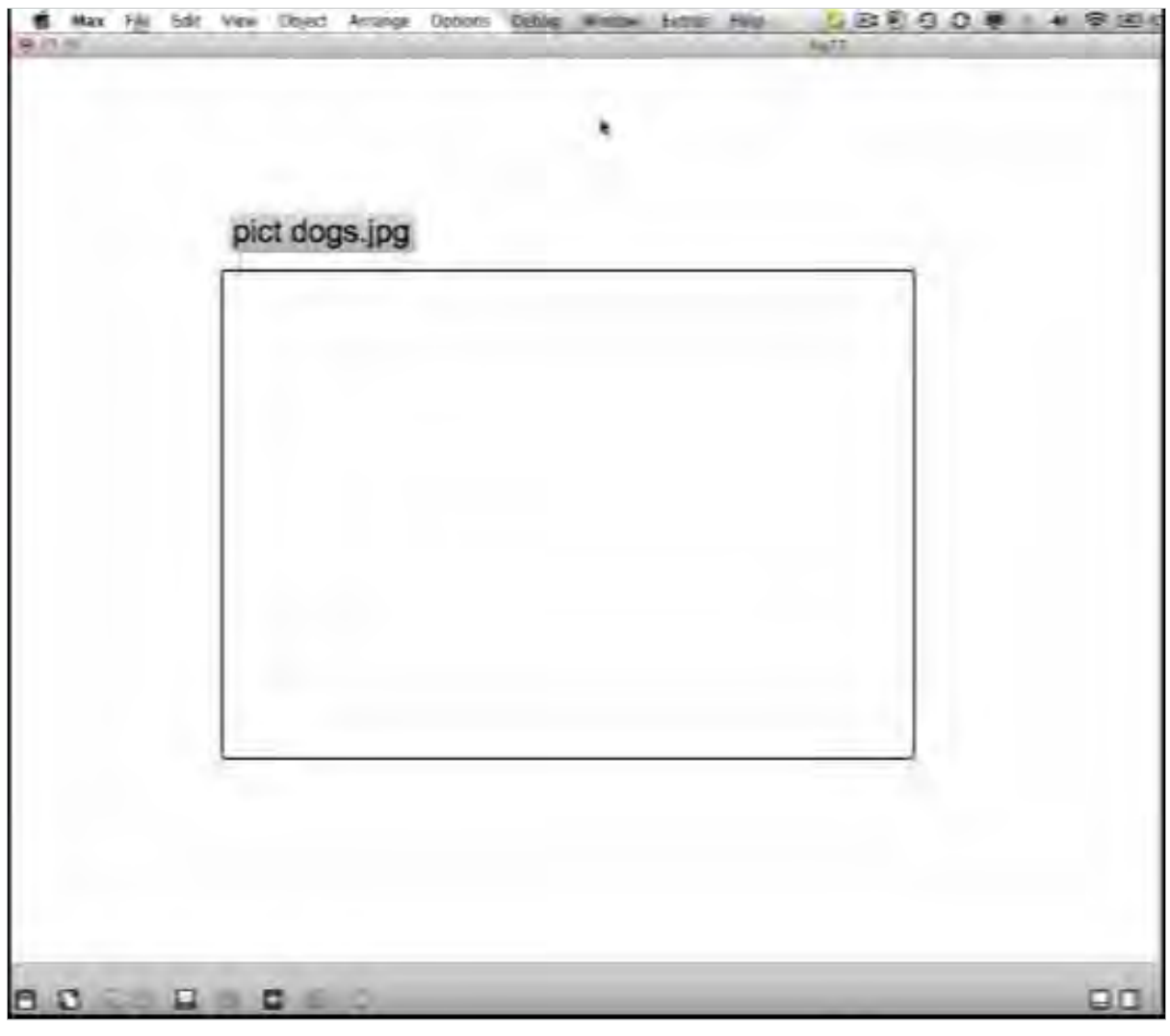


Figura 28